Name : Jadhav Nilesh Balu.
Class : TE-(B)
Roll No.: COTB012
Assignment No.: 12(DSBDA)

Program code :

```
package com.org.dsbda.weather;


import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.StringTokenizer;


import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.KeyValueTextInputFormat;

import org.apache.hadoop.mapred.MapReduceBase;
```

```
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class Weather extends Configured implements Tool {
     final long DEFAULT_SPLIT_SIZE = 128 * 1024 * 1024;


     public static class MapClass extends MapReduceBase
               implements Mapper<LongWritable, Text, Text, Text>
{

          private Text word = new Text();
          private Text values = new Text();

          public void map(LongWritable key, Text value,
                              OutputCollector<Text, Text>
output,
                              Reporter reporter) throws
IOException {
               String line = value.toString();
```

```
                StringTokenizer itr = new StringTokenizer(line);
                int counter = 0;
                String key_out = null;
                String value_str = null;
                boolean skip = false;
                loop:while (itr.hasMoreTokens() && counter<13) {
                     String str = itr.nextToken();
                     switch (counter) {
                     case 0:
                          key_out = str;
                          if(str.contains("STN")){
                               skip = true;
                               break loop;
                          }else{
                               break;
                          }
                     case 2:
                          int hour =
Integer.valueOf(str.substring(str.lastIndexOf("_")+1,
str.length()));
                          str =
str.substring(4,str.lastIndexOf("_")-2);



                          if(hour>4 && hour<=10){
```

```
                    str = str.concat("_section1");
                }else if(hour>10 && hour<=16){
                    str = str.concat("_section2");
                }else if(hour>16 && hour<=22){
                    str = str.concat("_section3");
                }else{
                    str = str.concat("_section4");
                }


                key_out =
key_out.concat("_").concat(str);
                break;
            case 3:
                if(str.equals("9999.9")){
                    skip = true;
                    break loop;
                }else{
                    value_str = str.concat(" ");
                    break;
                }
            case 4:
                if(str.equals("9999.9")){
                    skip = true;
```

```
                                        break loop;

                                   }else{

                                        value_str =
value_str.concat(str).concat(" ");

                                        break;

                                   }

                              case 12:

                                   if(str.equals("999.9")){

                                        skip = true;

                                        break loop;

                                   }else{

                                        value_str =
value_str.concat(str).concat(" ");

                                        break;

                                   }

                              default:

                                   break;

                              }

                              counter++;

                         }

                         if(!skip){

                              word.set(key_out);

                              values.set(value_str);

                              output.collect(word, values);
```

```
                }

            }

        }


        public static class MapClassForJob2 extends MapReduceBase

                implements Mapper<Text, Text, Text, Text> {

            private Text key_text = new Text();

            private Text value_text = new Text();

            @Override

            public void map(Text key, Text value,

                    OutputCollector<Text, Text> output, Reporter
reporter) throws IOException {



                String str = key.toString();

                String station =
str.substring(str.lastIndexOf("_")+1, str.length());

                str = str.substring(0,str.lastIndexOf("_"));

                key_text.set(str);



                StringTokenizer itr = new
StringTokenizer(value.toString());

                String str_out = station.concat("<");

                while (itr.hasMoreTokens()) {

                    String nextToken = itr.nextToken(" ");

                    str_out = str_out.concat(nextToken);
```

```
                    str_out = ((itr.hasMoreTokens()) ?
str_out.concat(",") : str_out.concat(">"));

                }

                value_text.set(str_out);

                output.collect(key_text,value_text);

          }

     }


     public static class Reduce extends MapReduceBase

                implements Reducer<Text, Text, Text, Text> {

          private Text value_out_text = new Text();

          public void reduce(Text key, Iterator<Text> values,

                    OutputCollector<Text, Text> output, Reporter
reporter) throws IOException {

                double sum_temp = 0;

                double sum_dew = 0;

                double sum_wind = 0;

                int count = 0;


                while (values.hasNext()) {


                    String str = values.next().toString();


                    StringTokenizer itr = new
StringTokenizer(str);
```

```
                    int count_vector = 0;
                    while (itr.hasMoreTokens()) {
                         String nextToken = itr.nextToken(" ");
                         if(count_vector==0){
                              sum_temp +=
Double.valueOf(nextToken);
                         }
                         if(count_vector==1){
                              sum_dew +=
Double.valueOf(nextToken);
                         }
                         if(count_vector==2){
                              sum_wind +=
Double.valueOf(nextToken);
                         }
                         count_vector++;
                    }
                    count++;
               }
               double avg_tmp = sum_temp / count;
               double avg_dew = sum_dew / count;
               double avg_wind = sum_wind / count;

               System.out.println(key.toString()+" count is
"+count+" sum of temp is "+sum_temp+" sum of dew is "+sum_dew+"
sum of wind is "+sum_wind+"\n");
```

```
                String value_out =
String.valueOf(avg_tmp).concat("
").concat(String.valueOf(avg_dew)).concat("
").concat(String.valueOf(avg_wind));

                value_out_text.set(value_out);

                output.collect(key, value_out_text);

          }

     }



     public static class ReduceForJob2 extends MapReduceBase

                implements Reducer<Text, Text, Text, Text> {

          private Text value_out_text = new Text();

          public void reduce(Text key, Iterator<Text> values,

                     OutputCollector<Text, Text> output, Reporter
reporter) throws IOException {

                String value_out = "";

                while (values.hasNext()) {

                     value_out =
value_out.concat(values.next().toString()).concat(" ");

                }

                value_out_text.set(value_out);

                output.collect(key, value_out_text);

          }

     }
```

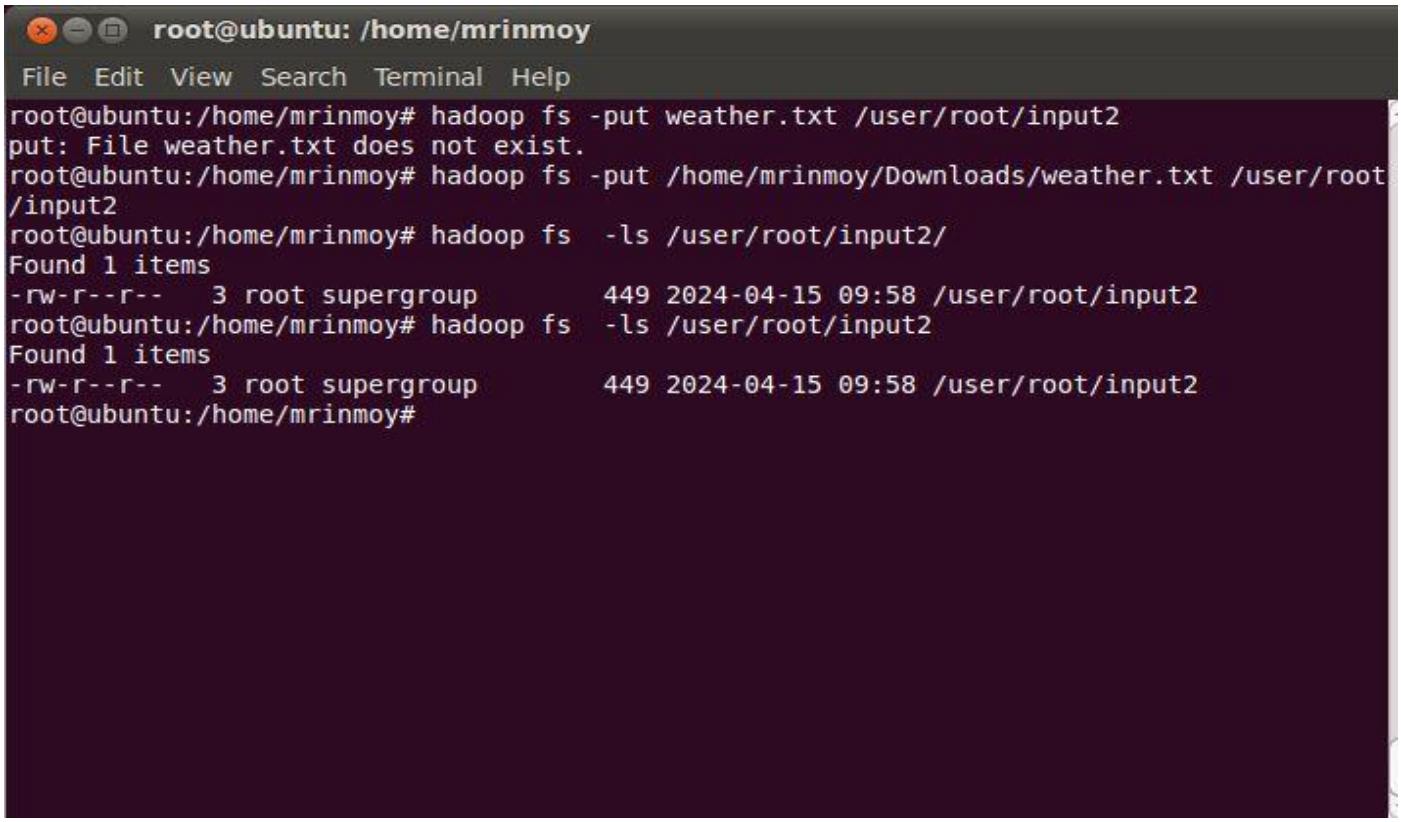
```
     static int printUsage() {

          System.out.println("weather [-m <maps>] [-r <reduces>]
<job_1 input> <job_1 output> <job_2 output>");

          ToolRunner.printGenericCommandUsage(System.out);

          return -1;

     }




     public int run(String[] args) throws Exception {

          Configuration config = getConf();




          JobConf conf = new JobConf(config, Weather.class);

          conf.setJobName("Weather Job1");




          conf.setOutputKeyClass(Text.class);


          conf.setOutputValueClass(Text.class);


          conf.setMapOutputKeyClass(Text.class);

          conf.setMapOutputValueClass(Text.class);


          conf.setMapperClass(MapClass.class);
```

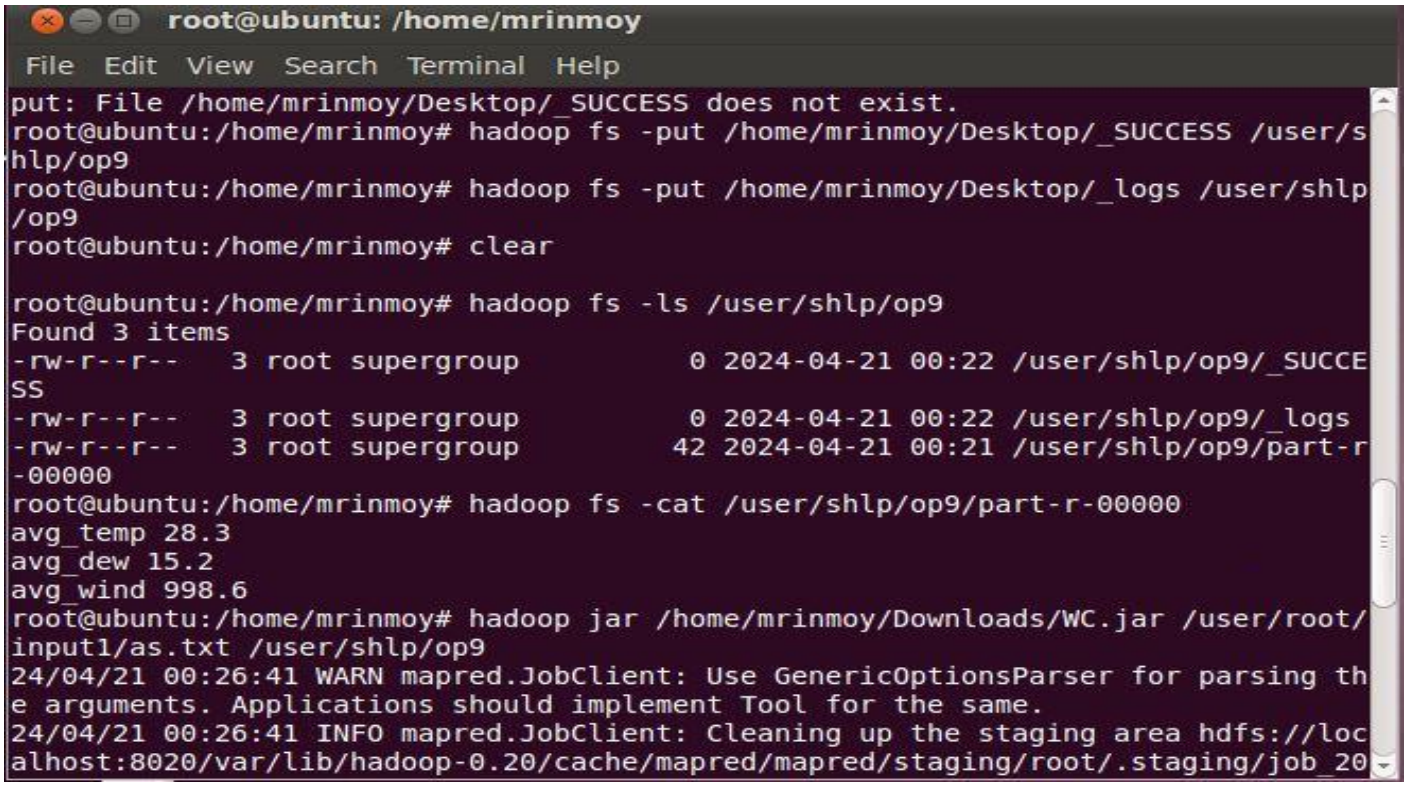
```
        conf.setReducerClass(Reduce.class);
        List<String> other_args = new ArrayList<String>();
        for(int i=0; i < args.length; ++i) {
            try {
                if ("-m".equals(args[i])) {
    conf.setNumMapTasks(Integer.parseInt(args[++i]));
                } else if ("-r".equals(args[i])) {
    conf.setNumReduceTasks(Integer.parseInt(args[++i]));
                } else {
                    other_args.add(args[i]);
                }
            } catch (NumberFormatException except) {
                System.out.println("ERROR: Integer expected
instead of " + args[i]);
                return printUsage();
            } catch (ArrayIndexOutOfBoundsException except) {
                System.out.println("ERROR: Required
parameter missing from " +
                                   args[i-1]);
                return printUsage();
            }
        }
```

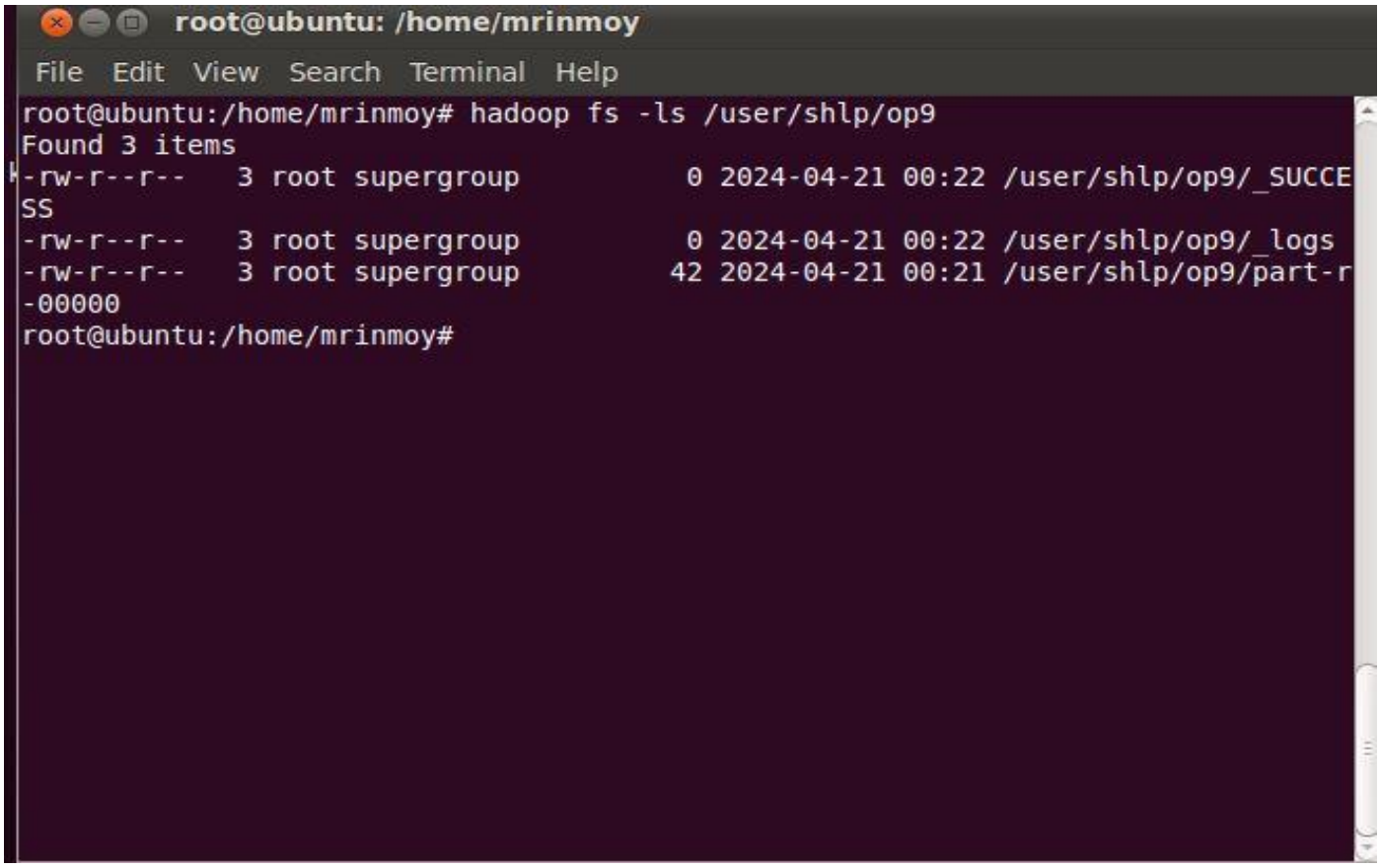
```
          FileInputFormat.setInputPaths(conf,
other_args.get(0));

          FileOutputFormat.setOutputPath(conf, new
Path(other_args.get(1)));



          JobClient.runJob(conf);



          JobConf conf2 = new JobConf(config, Weather.class);

          conf2.setJobName("Weather Job 2");




          conf2.setOutputKeyClass(Text.class);



          conf2.setOutputValueClass(Text.class);



          conf2.setInputFormat(KeyValueTextInputFormat.class);



          conf2.setMapOutputKeyClass(Text.class);

          conf2.setMapOutputValueClass(Text.class);



          conf2.setMapperClass(MapClassForJob2.class);



          conf2.setReducerClass(ReduceForJob2.class);
```

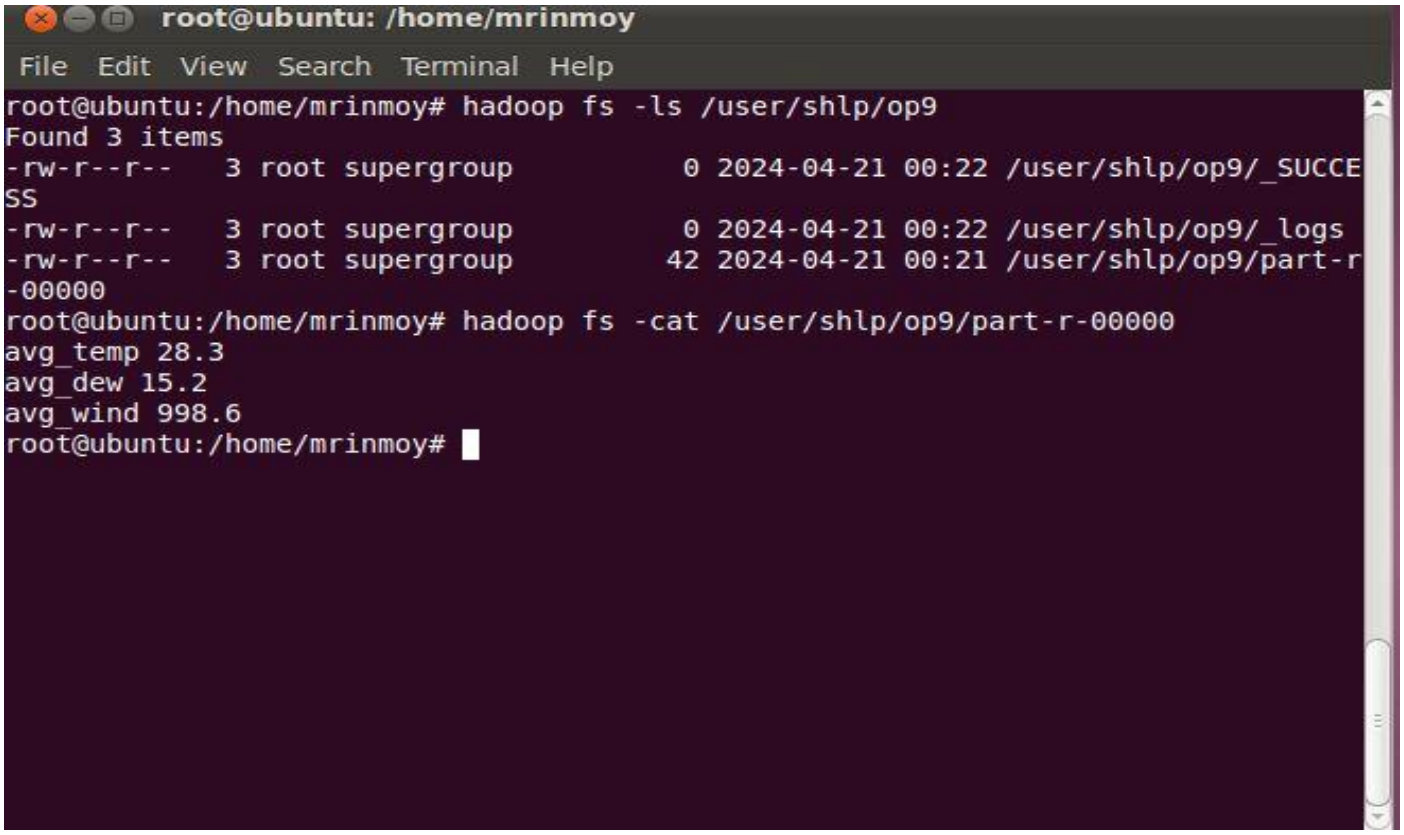
```
          FileInputFormat.setInputPaths(conf2, new
Path(other_args.get(1)));

          FileOutputFormat.setOutputPath(conf2, new
Path(other_args.get(2)));



          JobClient.runJob(conf2);

          return 0;

     }




     public static void main(String[] args) throws Exception {

          int res = ToolRunner.run(new Configuration(), new
Weather(), args);

          System.exit(res);

     }



}
```

Output:


root@ubuntu: /home/mrinmoy
File Edit View Search Terminal Help
root@ubuntu:/home/mrinmoy# hadoop fs -put weather.txt /user/root/input2
put: File weather.txt does not exist.
root@ubuntu:/home/mrinmoy# hadoop fs -put /home/mrinmoy/Downloads/weather.txt /user/root
/input2
root@ubuntu:/home/mrinmoy# hadoop fs -ls /user/root/input2/
Found 1 items
-rw-r--r-- 3 root supergroup 449 2024-04-15 09:58 /user/root/input2
root@ubuntu:/home/mrinmoy# hadoop fs -ls /user/root/input2
Found 1 items
-rw-r--r-- 3 root supergroup 449 2024-04-15 09:58 /user/root/input2
root@ubuntu:/home/mrinmoy#


root@ubuntu: /home/mrinmoy
File Edit View Search Terminal Help
put: File /home/mrinmoy/Desktop/_SUCCESS does not exist.
root@ubuntu:/home/mrinmoy# hadoop fs -put /home/mrinmoy/Desktop/_SUCCESS /user/s
hlp/op9
root@ubuntu:/home/mrinmoy# hadoop fs -put /home/mrinmoy/Desktop/_logs /user/shlp
/op9
root@ubuntu:/home/mrinmoy# clear
root@ubuntu:/home/mrinmoy# hadoop fs -ls /user/shlp/op9
Found 3 items
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_SUCCE
SS
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_logs
-rw-r--r-- 3 root supergroup 42 2024-04-21 00:21 /user/shlp/op9/part-r
-00000
root@ubuntu:/home/mrinmoy# hadoop fs -cat /user/shlp/op9/part-r-00000
avg_temp 28.3
avg_dew 15.2
avg_wind 998.6
root@ubuntu:/home/mrinmoy# hadoop jar /home/mrinmoy/Downloads/WC.jar /user/root/
input1/as.txt /user/shlp/op9
24/04/21 00:26:41 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
24/04/21 00:26:41 INFO mapred.JobClient: Cleaning up the staging area hdfs://loc
alhost:8020/var/lib/hadoop-0.20/cache/mapred/mapred/staging/root/.staging/job_20


root@ubuntu: /home/mrinmoy
File Edit View Search Terminal Help
root@ubuntu:/home/mrinmoy# hadoop fs -ls /user/shlp/op9
Found 3 items
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_SUCCESS
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_logs
-rw-r--r-- 3 root supergroup 42 2024-04-21 00:21 /user/shlp/op9/part-r-00000
root@ubuntu:/home/mrinmoy#


root@ubuntu: /home/mrinmoy
File Edit View Search Terminal Help
root@ubuntu:/home/mrinmoy# hadoop fs -ls /user/shlp/op9
Found 3 items
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_SUCCESS
-rw-r--r-- 3 root supergroup 0 2024-04-21 00:22 /user/shlp/op9/_logs
-rw-r--r-- 3 root supergroup 42 2024-04-21 00:21 /user/shlp/op9/part-r-00000
root@ubuntu:/home/mrinmoy# hadoop fs -cat /user/shlp/op9/part-r-00000
avg_temp 28.3
avg_dew 15.2
avg_wind 998.6
root@ubuntu:/home/mrinmoy#